**Hands on 3**

function x = f(n)

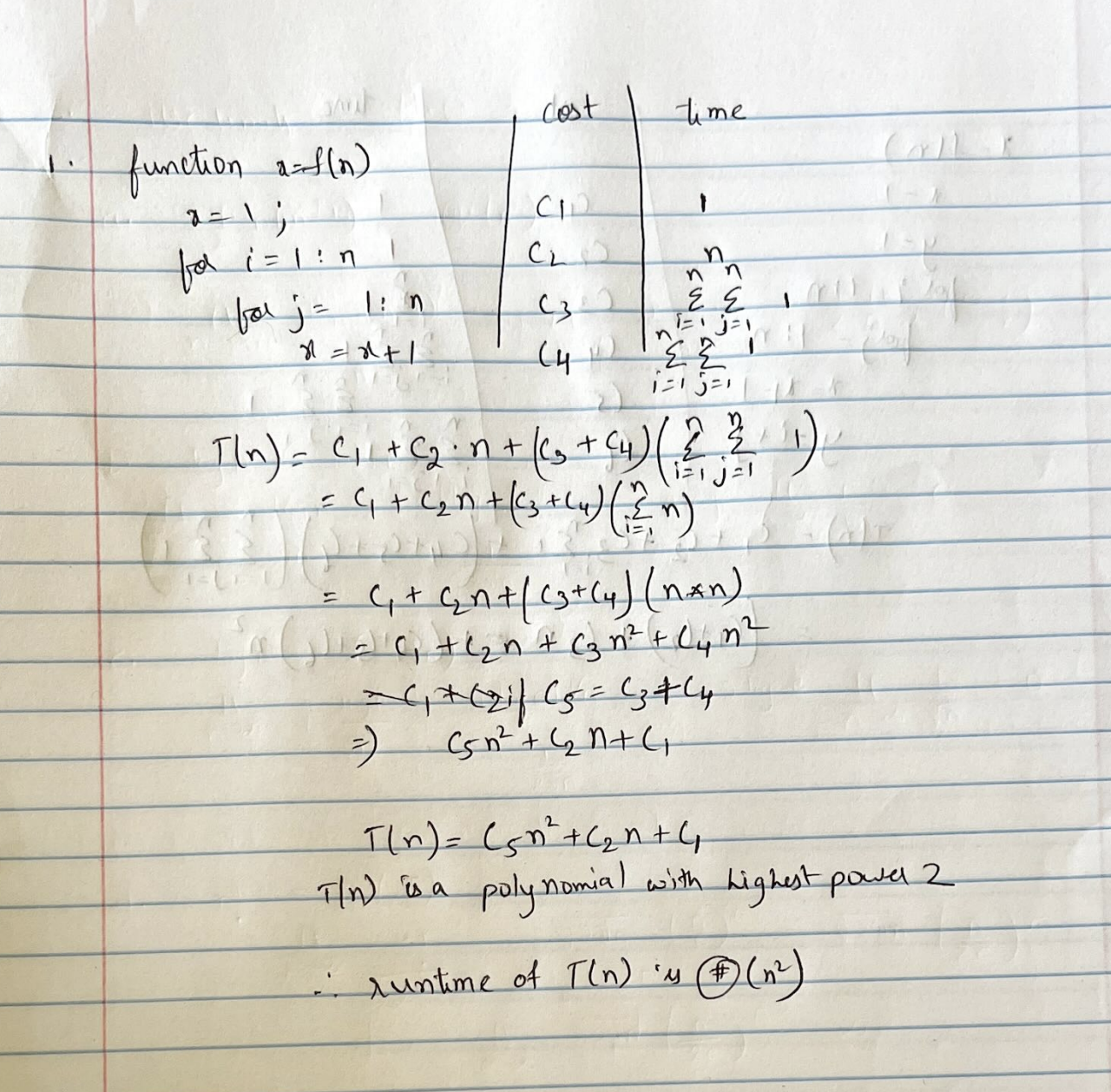
   x = 1;

   for i = 1:n

        for j = 1:n

             x = x + 1;

1**.Find the runtime of the algorithm mathematically**



**2.Time this function for various n e.g. n = 1,2,3.... You should have small values of n all the way up to large values. Plot "time" vs "n" (time on y-axis and n on x-axis). Also, fit a curve to your data, hint it's a polynomial.**

From 1. We get T(n) = C5 n2 + C2 n + C1    
assume C5. =1, C2 =3, C1= 1  
=> T(n) = n2+3n+1

A graph with a line drawn on it

Description automatically generated

**3. Find polynomials that are upper and lower bounds on your curve from #2. From this specify a big-O, a big-Omega, and what big-theta is.**

Considering T(n) = C5 n2 + C2 n + C1Upper bound : Cg (n) = 5 n2   
big -O is O(n2)

Considering lower bound as Cg(n) = 0.5n2  
Big Omega is Ω(n2)  
  
graphs for T(n), upper bound and lower bound

A graph of a line graph

Description automatically generated

For big theta we have upper bound as 5n2 , lower bound as 0.5n2 considering the definition of big theta Θ(g(n)) = { f(n): there exist positive constants c1, c2 and n0 such that 0 ≤ c1g(n) ≤ f(n) ≤ c2g(n) for all n ≥ n0 } we can assume big theta as Θ(n2)

**4. Find the approximate (eye ball it) location of "n\_0" . Do this by zooming in on your plot and indicating on the plot where n\_0 is and why you picked this value. Hint: I should see data that does not follow the trend of the polynomial you determined in #2.**

A graph of a graph with a line

Description automatically generated with medium confidence

From this we can say that “n\_0” is 1 because for all values of n which are greater than or equal to 1. For all n>1 we get 5n2 <= T(n) <= 0.5n2

If I modified the function to be:

x = f(n)

   x = 1;

   y = 1;

   for i = 1:n

        for j = 1:n

             x = x + 1;

        y = i + j;

**4. Will this increate how long it takes the algorithm to run (e.x. you are timing the function like in #2)?**

Yes, the modification introduces an additional operation inside the nested loop, increasing the overall running time.

A paper with writing on it

Description automatically generated

**5. Will it effect your results from #1?**

No, Overall time complexity will remain same O(n2), the number of steps in the function increased, as observed only change is in the constant value and not the structure of T(n)